

# ON DEGREES OF MODULAR COMMON DIVISORS AND THE BIG PRIME GCD ALGORITHM

VAHAGN H. MIKAELIAN  
E-mail: v.mikaelian@gmail.com

**ABSTRACT.** We consider a few modifications of the Big prime modular **gcd** algorithm for polynomials in  $\mathbb{Z}[x]$ . Our modifications are based on bounds of degrees of modular common divisors of polynomials, on estimates of the number of prime divisors of a resultant and on finding preliminary bounds on degrees of common divisors using auxiliary primes. These modifications are used to suggest improved algorithms for **gcd** calculation and for coprime polynomials detection. To illustrate the ideas we apply the constructed algorithms on certain polynomials, in particular, on polynomials from Knuth's example of intermediate expression swell.

## CONTENTS

1. Introduction	1
2. The <b>gcd</b> in polynomial rings and the degrees of common divisors	3
3. Some notations for modular reductions	7
4. Problems at lifting the modular <b>gcd</b> to $\mathbb{Z}[x]$	8
4.1. Problem 1. Avoiding the eliminating coefficients	8
4.2. Problem 2. Negative coefficients and reconstruction of the pre-image	9
4.3. Problem 3. Finding the correct fold of the modular <b>gcd</b> of right degree	10
4.4. Problem 4. Finding the right degree for the modular <b>gcd</b>	11
5. The Big prime modular <b>gcd</b> algorithm	13
6. Estimating the prime divisors of the resultant	15
7. An algorithm to check coprime polynomials	16
8. Other modifications of algorithms	18
References	22

## 1. INTRODUCTION

This work is one of the articles in which we would like to present parts from new Introduction to computer algebra [11], that currently is under preparation. In [11] we try to give a “more algebraic” and detailed view on some of the areas of computer algebra, such as, algorithms on Euclidean rings, extensions of fields, operators in spaces on finite fields, factorization in UFD's, etc..

---

*Date:* July 23, 2014.

*1991 Mathematics Subject Classification.* 20E10, 20E15, 20E22, 20F16.

*Key words and phrases.* Big prime modular gcd algorithm, Euclidean rings, Euclidean algorithm, UFD, greatest common divisor, algorithm, computer algebra, ring theory, modular rings, finite fields.

The author was supported in part by SCS RA, joint Armenian-Russian research project 13RF-030 and by State Committee Science MES RA grant in frame of project 13-1A246.

The Big prime modular **gcd** algorithm is one of the first and most popular algorithms of computer algebra. In its classical form it allows to calculate the greatest common divisor  $\mathbf{gcd}(f(x), g(x))$  for any non-zero polynomials  $f(x), g(x) \in \mathbb{Z}[x]$ . There are a few modifications of this algorithm for other UFD's, such as multivariate polynomial rings. Attention to the **gcd** calculation is partially explained by the first examples that were built to explain importance of application of algebraic methods to computer science. In particular, Knuth's well known example of intermediate expression swell discusses the polynomials

$$(1) \quad \begin{aligned} f(x) &= x^8 + x^6 - 3x^4 - 3x^3 + 8x^2 + 2x - 5, \\ g(x) &= 3x^6 + 5x^4 - 4x^2 - 9x + 21, \end{aligned}$$

and it shows that calculation of  $\mathbf{gcd}(f(x), g(x))$  by traditional Euclidean algorithm on rational numbers generates very large integers to deal with, whereas consideration of these polynomials modulo  $p$ , that is, consideration of their images under ring homomorphism  $\varphi_p : \mathbb{Z}[x] \rightarrow \mathbb{Z}_p[x]$  (where  $\mathbb{Z}_p[x]$  is the polynomial ring over the residue ring  $\mathbb{Z}_p \cong \mathbb{Z}/p\mathbb{Z}$ ) very easily shows that  $\mathbf{gcd}(f(x), g(x)) = 1$  (see [7] and also [1, 15, 4, 14]). We are going to use the polynomials (1) as examples below to apply the algorithms below (see examples 5.2, 6.2, 6.6, 7.2, 8.1, 8.3).

The main idea of the Big prime modular **gcd** algorithm is that for the given polynomials  $f(x), g(x) \in \mathbb{Z}[x]$  one may first consider their images  $f_p(x) = \varphi_p(f(x)), g_p(x) = \varphi_p(g(x)) \in \mathbb{Z}_p[x]$  under  $\varphi_p$ . Unlike  $\mathbb{Z}[x]$ , the ring  $\mathbb{Z}_p[x]$  is an Euclidean domain, since it is a polynomial ring over a field, so the  $\mathbf{gcd}(f_p(x), g_p(x))$  can be computed in it by the well known Euclidean algorithm. There remains “to lift” a certain fold  $t \cdot \mathbf{gcd}(f_p(x), g_p(x))$  of it to the ring  $\mathbb{Z}[x]$  to reconstruct the pre-image  $\mathbf{gcd}(f(x), g(x))$ . The “lifting” procedure consists of selecting the suitable value for prime  $p$ , then finding in  $\mathbb{Z}[x]$  an appropriate pre-image for  $\mathbf{gcd}(f_p(x), g_p(x))$ , then checking if that pre-image divides both  $f(x)$  and  $g(x)$ . If yes, it is the  $\mathbf{gcd}(f(x), g(x))$  we are looking for. If not, then a new  $p$  need be selected to repeat the process. Arguments based on resultants and on Landau-Mignotte bounds show that we can effectively choose  $p$  such that the number of required repetitions is “small”.

The first aim of this work is to present in Sections 2–5 a slightly modified argumentation of the algorithm, based on comparison of the degrees of common divisors of  $f(x)$  and  $g(x)$  in  $\mathbb{Z}[x]$ , and of  $f_p(x)$  and  $g_p(x)$  in  $\mathbb{Z}_p[x]$  (see Algorithm 5.1). This approach allows some simplification of a step of the algorithm: for some primes  $p$  we need not reconstruct the pre-image of  $t \cdot \mathbf{gcd}(f_p(x), g_p(x))$ , but we immediately get an indication that this prime is not suitable, and we should proceed to a new  $p$  (see Remark 5.1).

Then in Section 6 we discuss the problem if or not the Big prime modular **gcd** algorithm could output the correct answer using just one prime  $p$ . The answer is positive, but for some reasons it should not be used to improve the algorithm (to make it work with one  $p$ ) because it evolves a too large prime (see Remark 6.3). Instead, we show that we can estimate the maximal number of  $p$ 's (repetitions of steps) that may be used in traditional Big prime modular **gcd** algorithm. For example, for the polynomials (1) of Knuth's example this number is at most 31. Estimates of this type can be found in literature elsewhere. We just make the bound considerably smaller (see Remark 6.7).

The obtained bounds on the number of primes  $p$  are especially effective when we are interested not in **gcd**, but just in detection if the polynomials  $f(x), g(x) \in \mathbb{Z}[x]$  are coprime or not. We consider this in Section 7 (see Algorithm 7.1).

In Section 8 we consider four other ideas to modify the Big prime modular **gcd** algorithm. Two first ideas are based on checking the number of primes  $p$ . The third idea is based on using an auxiliary prime  $q$  to estimate the degree of **gcd** $(f(x), g(x))$  by means of the degree of **gcd** $(f_q(x), g_q(x))$  (see Algorithm 8.1). Example 8.1 shows how much better results we may get by this modification. The fourth idea combines both approaches: it uses a set of auxiliary primes  $q_1, \dots, q_{k+1}$  to correctly find the degree of **gcd** $(f(x), g(x))$ , and then we use a modified version of Landau-Mignotte bound to find a single big prime  $p$  by which we can calculate the **gcd** $(f_p(x), g_p(x))$ .

The arguments used here can be generalized for the case of polynomials on general UFD's. From the unique factorization in a UFD it easily follows, that **gcd** always exists, and it is easy to detect if or not the given common divisor of maximal degree is a **gcd** or not. The less simple part is to find ways to compute **gcd** (without having the prime-power factorization). That can be done for some classes of UFD's, such as, multivariate polynomials on fields. The case of general UFD's will be considered later [12].

## 2. THE **gcd** IN POLYNOMIAL RINGS AND THE DEGREES OF COMMON DIVISORS

The problem of finding the greatest common divisor **gcd** $(a, b)$  of any non-zero elements  $a, b$  in a ring  $R$  can be separated to two tasks:

- (1) finding out if **gcd** $(a, b)$ , in general, *exists* for  $a, b \in R$ ; and then:
- (2) finding an *effective* way to calculate the **gcd** $(a, b)$ .

The Euclidean algorithm gives an easy answer to both of these tasks in any Euclidean domain, that is, an integrity domain  $R$  possessing *Euclidean norm*  $\delta : R \setminus 0 \rightarrow \mathbb{N} \cup \{0\}$ , such that  $\delta(ab) \geq \delta(a)$  hold for any non-zero elements  $a, b \in R$ ; and for any  $a, b \in R$ , where  $b \neq 0$ , there exists elements  $q, r \in R$ , such that  $a = qb + r$ , where either  $r = 0$ , or  $r \neq 0$  and  $\delta(r) < \delta(b)$  [10, 5, 3, 8, 15, 4]. The Euclidean algorithm works for any polynomial ring  $K[x]$  over a field  $K$ , such as  $\mathbb{Q}[x]$ ,  $\mathbb{R}[x]$ ,  $\mathbb{C}[x]$ ,  $\mathbb{Z}_p[x]$  because these rings can easily be turned to an Euclidean domain by defining  $\delta(f(x)) = \deg f(x)$  for any non-zero  $f(x) \in K[x]$ .

The situation is less simple in non-Euclidean domains, even in such a widely used ring as the ring  $\mathbb{Z}[x]$  of polynomials with integer coefficients. That  $\mathbb{Z}[x]$  is not an Euclidean domain is easy to show by elements  $x, 2 \in \mathbb{Z}[x]$ . If  $\mathbb{Z}[x]$  were an Euclidean domain, it would contain elements  $u(x), v(x)$  such that  $x \cdot u(x) + 2 \cdot v(x) = \mathbf{gcd}(x, 2) = \pm 1$ , which is not possible.

The first of two tasks mentioned above, namely, *existence* of **gcd** can be accomplished for  $\mathbb{Z}[x]$  by proving that  $\mathbb{Z}[x]$  is a UFD, that is, an integrity domain in which every non-zero element  $a$  has a factorization  $a = \epsilon p_1 \cdots p_k$ , where  $\epsilon \in R^*$  is a unit (invertible) element in  $R$ , the elements  $p_i$  are prime for all  $i = 1, \dots, k$ , and where the factorization above is unique in the sense that if  $a$  has another factorization of

that type  $\theta q_1 \cdots q_s$ , where  $\theta \in R^*$  and the elements  $q_i$  are prime, then  $k = s$  and (perhaps after some reordering of the prime factors) the respective prime elements are associated:  $p_i \approx q_i$  for all  $i = 1, \dots, k$ . For brevity, in the sequel we will often omit the phrase “perhaps after some reordering of the prime factors” and this will cause no confusion.

After merging the associated prime elements together, we get a unique factorization into prime-power elements:

$$(2) \quad a = \nu p_1^{\alpha_1} \cdots p_n^{\alpha_n}, \quad \nu \in R^*, \quad \alpha_i \in \mathbb{N} \text{ and } p_i \not\approx p_j \text{ for any } i \neq j; \quad i, j = 1, \dots, n$$

(in some arguments below we may admit some of the factors  $p_i^{\alpha_i}$  participate with degrees  $\alpha_i = 0$ , this makes some notations simpler). From this it is easy to see that in a UFD  $R$  the  $\mathbf{gcd}(a, b)$  exists for any non-zero elements  $a, b \in R$ . Assume  $b \in R$  has the factorization

$$b = \kappa p_1^{\alpha'_1} \cdots p_n^{\alpha'_n}, \quad \kappa \in R^*$$

(we use the same primes  $p_i$  in both factorizations because if, say,  $p_i$  is not actually participating in one of those factorizations, we can add it as  $p_i^{\alpha_i}$  with  $\alpha_i = 0$ ). Then

$$(3) \quad \mathbf{gcd}(a, b) = d = p_1^{\gamma_1} \cdots p_n^{\gamma_n},$$

where  $\gamma_i = \min\{\alpha_i, \alpha'_i\}$ . This follows from uniqueness of factorization in UFD. For, if  $h$  is a common divisor of  $a, b$ , and if  $p_i$  is a prime divisor of  $h$ , then it also is a prime divisor of  $a$  and of  $b$ . The elements  $p_i$  cannot participate in factorization of  $h$  by a power greater than  $\min\{\alpha_i, \alpha'_i\}$ , because then  $a$  (or  $b$ ) would have an alternative factorization in which  $p_i$  occurs more than  $\alpha_i$  (or  $\alpha'_i$ ) times.

The shortest way to see that  $\mathbb{Z}[x]$  is a UFD is to apply Gauss's Theorem: if the ring  $R$  is a UFD, then the polynomial ring  $R[x]$  also is a UFD [10, 5, 2, 8, 15]. Since  $\mathbb{Z}$  is a UFD (that fact is known as “the fundamental theorem of arithmetic”),  $\mathbb{Z}[x]$  also is a UFD.

Clearly,  $\mathbf{gcd}(a, b)$  is defined up to a unit multiplier from  $R^*$ . For integers from  $R = \mathbb{Z}$  or for polynomials from  $R = \mathbb{Z}[x]$  this unit multiplier can be just  $-1$  or  $1$ . So to say:  $\mathbf{gcd}(a, b)$  is defined “up to the sign  $\pm 1$ ” because  $\mathbb{Z}^* = \mathbb{Z}[x]^* = \{-1, 1\}$ . And for polynomials from  $R = \mathbb{Z}_p[x]$  the  $\mathbf{gcd}(a, b)$  is defined up to any non-zero multiplier  $t \in \mathbb{Z}_p^* = \{1, \dots, p-1\}$ . Taking this into account we can use  $\mathbf{gcd}(a, b) = 1$  and  $\mathbf{gcd}(a, b) \approx 1$  as equivalent notations, since associated elements are defined up to a unit multiplier. Notice that in some sources they prefer to additionally introduce a normal form of the  $\mathbf{gcd}$  to distinguish one fixed instance of the  $\mathbf{gcd}$ . Instead of using that extra term, we will just in a few places refer to the “positive  $\mathbf{gcd}$ ”, meaning that we take, say,  $2 = \mathbf{gcd}(6, 8)$ , and not  $-2$ .

Furthermore, since the content  $\text{cont}(f(x))$  of a polynomial  $f(x)$  is a  $\mathbf{gcd}$  for some elements (coefficients of the polynomials), the constant and the primitive part  $\text{pp}(f(x)) = f(x)/\text{cont}(f(x))$  can also be considered up to a unit multiplier. For a non-zero polynomial  $f(x) \in \mathbb{Z}[x]$  we can choose the  $\text{cont}(f(x))$  so that  $\text{sgn } \text{cont}(f(x)) = \text{sgn } \text{lc}(f(x))$ , that is, the  $\text{cont}(f(x))$  has the same sign as the leading coefficient of  $f(x)$ . Then the leading coefficient  $\text{lc}(\text{pp}(f(x)))$  of the primitive part  $\text{pp}(f(x)) = f(x)/\text{cont}(f(x))$  will be positive. We will use this below without special notification.

Now we would like to a little restrict the algebraic background we use. Two main algebraic systems, used in the Big prime modular **gcd** algorithm, are the Euclidean domains and the UFD's. However, their usage is "asymetric" in the sense that the Euclidean domains and Euclidean algorithm are used in many parts of the Big prime modular **gcd** algorithm, whereas the UFD's are used just to prove that **gcd** does exist. Moreover, it is easy to understand that (2) and (3) may hardly be effective tools to calculate a **gcd**, since they are using factorization of elements to primes, while finding such a factorization is a more complicated task than finding just the **gcd**. Thus, it is reasonable to drop the UFD's from consideration, and to obtain (2) directly using Gauss's Lemma on primitive polynomials in  $\mathbb{Z}[x]$  (a polynomial  $f(x) \in \mathbb{Z}[x]$  is primitive if  $\text{cont}(f(x)) \approx 1$ , that is,  $\text{pp}(f(x)) = f(x)/\text{cont}(f(x)) \approx f(x)$ ).

By Gauss's Lemma, a product of two primitive polynomials is primitive in  $\mathbb{Z}[x]$  [10, 5, 2, 8, 15]. So if

$$(4) \quad f(x) = \text{cont}(f(x)) \cdot \text{pp}(f(x)) \text{ and } g(x) = \text{cont}(g(x)) \cdot \text{pp}(g(x)),$$

then

$$(5) \quad \begin{aligned} \text{cont}(f(x) \cdot g(x)) &= \text{cont}(f(x)) \cdot \text{cont}(g(x)) \\ \text{pp}(f(x) \cdot g(x)) &= \text{pp}(f(x)) \cdot \text{pp}(g(x)). \end{aligned}$$

The following is easy to deduce from Gauss's Lemma

**LEMMA 2.1.** *If  $f(x), t(x) \in \mathbb{Z}[x]$  and  $t(x)$  is primitive, then if  $t(x)$  divides  $f(x)$  in the ring  $\mathbb{Q}[x]$ , then  $t(x)$  also divides  $f(x)$  in  $\mathbb{Z}[x]$ .*

The unique factorization of any non-zero  $f(x) \in \mathbb{Z}[x]$  is easy to obtain from decompositions (5) above and from Lemma 2.1. Let us just outline it, the details can be found in [10, 5, 15, 4, 11]. By the fundamental theorem of arithmetic  $\text{cont}(f(x))$  can in a unique way be presented as a products of powers of primes:  $\text{cont}(f(x)) = \nu p_1^{\alpha_1} \cdots p_n^{\alpha_n}$ . So, if  $\deg f(x) = 0$ , then we are done.

Assume  $\deg f(x) > 0$ . If  $f(x)$  is not prime, then by repeatedly splitting it to products of factors of lower degree as many times as needed, we will eventually get a presentation of  $f(x)$  as a product of  $\text{cont}(f(x))$  and of some finitely many primitive prime polynomials  $q_i(x)$  of degrees greater than 0. We do not yet have the uniqueness of this decomposition, but we can still group the associated elements together to get the presentation:

$$(6) \quad f(x) = \text{cont}(f(x)) \cdot \text{pp}(f(x)) = \nu p_1^{\alpha_1} \cdots p_n^{\alpha_n} \cdot q_1^{\beta_1}(x) \cdots q_m^{\beta_m}(x).$$

If  $f(x)$  has another, alternative presentation of this sort and if  $t(x)$  is one of the primitive prime factors (of degree greater than 0) of that presentation, then the product (6) is divisible by  $t(x)$ . By Lemma 2.1,  $t(x)$  divides  $f(x)$  also in  $\mathbb{Z}[x]$ . Since  $t(x)$  is prime, it is associated to one of  $q_i(x)$ . Eliminate one instance of this  $q_i(x)$  in (6) and consider  $f(x)/q_i(x)$ . If  $f(x)/q_i(x)$  also is divisible by  $q_i(x)$ , we repeat the process. If not, we turn to other primitive prime polynomials (of degree greater than 0) dividing what remains from (6) after eliminations. After finitely many steps (6) will become  $\nu p_1^{\alpha_1} \cdots p_n^{\alpha_n}$ , and also from the other, alternative presentation a constant should be left only. So we apply the fundamental theorem of arithmetic one more time to get that (6) is the unique factorization.

We see that (6) is a particular case of (2). The proof above avoided usage of Gauss's Theorem and the formal definitions of the UFD's. And we see that the prime elements of  $\mathbb{Z}[x]$  are of two types: *prime numbers* and *primitive prime polynomials* of degrees greater than 0.

Existence of  $\mathbf{gcd}(f(x), g(x))$  for any two non-zero polynomials in  $f(x), g(x) \in \mathbb{Z}[x]$  can be deduced from (6) in analogy with (3). If

$$(7) \quad g(x) = \nu' p_1^{\alpha'_1} \cdots p_n^{\alpha'_n} \cdot q_1^{\beta'_1}(x) \cdots q_m^{\beta'_m}(x),$$

then

$$(8) \quad \mathbf{gcd}(f(x), g(x)) = \kappa p_1^{\gamma_1} \cdots p_n^{\gamma_n} \cdot q_1^{\delta_1}(x) \cdots q_m^{\delta_m}(x),$$

where  $\kappa = \pm 1$ ,  $\gamma_i = \min\{\alpha_i, \alpha'_i\}$ ,  $\delta_j = \min\{\beta_j, \beta'_j\}$ ; ( $i = 1, \dots, n$ ;  $j = 1, \dots, m$ ). However, like we admitted earlier, (3) and (8) are no effective tools to calculate the  $\mathbf{gcd}$ . We will turn to  $\mathbf{gcd}$  calculation algorithm in the next section.

(3) and (8) allow us to get some information that we will be essential later. Observe that the following definition of  $\mathbf{gcd}$ , often used in elementary mathematics, is no longer true for general polynomial rings: “ $d(x)$  is the greatest common divisor of  $f(x)$  and  $g(x)$  if it is their common divisor of maximal degree”. For example, for  $f(x) = 12x^2 + 24x + 12$  and  $g(x) = 8x + 8$  the maximum of degrees of their common divisors is 1. Nevertheless,  $h(x) = x + 1$  is not the  $\mathbf{gcd}(f(x), g(x))$ , although  $h(x)|f(x)$ ,  $h(x)|g(x)$  and  $\deg h(x) = 1$ . For,  $x + 1$  is not divisible by the common divisor  $2x + 2$ . We can detect the cases when the divisor of highest degree is the  $\mathbf{gcd}$ .

LEMMA 2.2. *For polynomials  $f(x), g(x) \in \mathbb{Z}[x]$  their common divisor of maximal degree  $h(x)$  is their  $\mathbf{gcd}$  if and only if  $\text{cont}(h(x)) \approx \mathbf{gcd}(\text{cont}(f(x)), \text{cont}(g(x)))$ .*

The lemma easily follows from (6), (7) and (8). We see that in example above the condition was missing:  $\text{cont}(x + 1) = 1$  but  $\mathbf{gcd}(\text{cont}(f(x)), \text{cont}(g(x))) = \mathbf{gcd}(12, 8) \approx 4 \not\approx 1$ . In fact,  $\mathbf{gcd}(f(x), g(x)) \approx 4x + 4$ .

COROLLARY 2.3. *For primitive polynomials  $f(x), g(x) \in \mathbb{Z}[x]$  their common divisor of maximal degree  $h(x)$  is their  $\mathbf{gcd}$  if and only if  $h(x)$  is primitive.*

In the case if polynomials are over a field, the situation is simpler. For any field  $K$  the polynomial ring  $K[x]$  is a UFD (and even an Euclidean domain). Any non-zero  $f(x) \in K[x]$  has a factorization

$$(9) \quad f(x) = \theta \cdot q_1^{\beta_1}(x) \cdots q_m^{\beta_m}(x), \quad \theta \in K^*, \quad \deg q_i(x) > 0, \quad i = 1, \dots, m,$$

which is unique in the sense mentioned above. Since all non-zero scalars in  $K$  are units, what we in (6) above had as a product of some prime numbers, actually “merges” in  $K$  into a unit:

$$\nu \cdot p_1^{\alpha_1} \cdots p_n^{\alpha_n} = \theta \in K^* = K \setminus \{0\}.$$

Comparing factorizations of type (9) for any non-zero polynomials  $f(x), g(x) \in K[x]$  we easily get:

LEMMA 2.4. *For any non-zero polynomials  $f(x), g(x) \in K[x]$  over a field  $K$  their common divisor of maximal degree  $h(x)$  is their  $\mathbf{gcd}$ .*

This, in particular, is true for rings mentioned above:  $\mathbb{Q}[x]$ ,  $\mathbb{R}[x]$ ,  $\mathbb{C}[x]$ ,  $\mathbb{Z}_p[x]$ . We will use this fact later to construct the Big prime modular **gcd** algorithm and its modifications.

The analog of Lemma 2.4 was not true for  $\mathbb{Z}[x]$  because in factorization (8) we have the non-unit prime-power factors  $p_i^{\gamma_i}$  which do participate in factorization of  $d(x) = \mathbf{gcd}(f(x), g(x))$ , but which add *nothing* to the degree of  $d(x)$ . This is why maximality of the degree is no longer the only criterion in  $\mathbb{Z}[x]$  to detect if the given  $h(x)$  is the **gcd** or not.

### 3. SOME NOTATIONS FOR MODULAR REDUCTIONS

The following notations, adopted from [11], are to make our arguments shorter and more uniform when we deal with numerals, polynomials and matrices. As above, let  $\mathbb{Z}_p$  be the residue ring (finite Galois field  $\mathbb{Z}_p = \mathbb{F}_p = \{0, \dots, p-1\}$ ) and

$$\varphi_p : \mathbb{Z} \rightarrow \mathbb{Z}_p$$

be the rings homomorphism mapping each  $z \in \mathbb{Z}$  to the remainder after division of  $z$  by  $p$ . That is,  $\varphi_p(z) \equiv z \pmod{p}$ , and  $\varphi_p(z) \in \mathbb{Z}_p$ .

We use the same symbol  $\varphi_p$  to denote the homomorphism

$$\varphi_p : \mathbb{Z}[x] \rightarrow \mathbb{Z}_p[x],$$

where  $\mathbb{Z}_p[x]$  is the ring of polynomials over  $\mathbb{Z}_p$ , and  $\varphi_p$  is mapping each of the coefficients  $a_i$  of  $f(x) \in \mathbb{Z}[x]$  to the remainder after division of  $a_i$  by  $p$ .

Similarly, we define the homomorphism of matrix rings

$$\varphi_p : M_{m,n}(\mathbb{Z}) \rightarrow M_{m,n}(\mathbb{Z}_p),$$

which maps each of the elements  $a_{ij}$  of a matrix  $A \in M_{m,n}(\mathbb{Z})$  to the remainder after division of  $a_{ij}$  by  $p$ .

Using the same symbol  $\varphi_p$  for numeric, polynomial and matrix homomorphisms causes no misunderstanding below, and it is more comfortable for some reasons. These homomorphisms are called “*modular reductions*” or just “*reductions*”. We can also specify these homomorphism as: “*numeric modular reduction*”, “*polynomial modular reduction*” or “*matrix modular reduction*” where needed [11].

For  $a \in \mathbb{Z}$  denote  $\varphi_p(a) = a_p$ . For  $f(x) \in \mathbb{Z}[x]$  denote  $\varphi_p(f(x)) = f_p(x) \in \mathbb{Z}_p[x]$ . So if

$$(10) \quad f(x) = a_0x^n + \dots + a_n$$

then:

$$f_p(x) = \varphi_p(f(x)) = \varphi_p(a_0)x^n + \dots + \varphi_p(a_n) = a_{0,p}x^n + \dots + a_{n,p} \in \mathbb{Z}_p[x].$$

And for a matrix  $A \in M_{m,n}(\mathbb{Z})$  denote  $\varphi_p(A) = A_p \in M_{m,n}(\mathbb{Z}_p)$ . If  $A = \|a_{i,j}\|_{m \times n}$  then  $A_p = \|\varphi_p(a_{i,j})\|_{m \times n} = \|a_{i,j,p}\|_{m \times n}$ .

#### 4. PROBLEMS AT LIFTING THE MODULAR **gcd** TO $\mathbb{Z}[x]$

Now we turn to the second task mentioned earlier: *effective calculation* of the actual  $\mathbf{gcd}(f(x), g(x))$  for the given non-zero polynomials  $f(x), g(x) \in \mathbb{Z}[x]$ .

The ring  $\mathbb{Z}_p[x]$  is an Euclidean domain, unlike the ring  $\mathbb{Z}[x]$ . So we can use the Euclidean algorithm to calculate **gcd** for any non-zero polynomials in  $\mathbb{Z}_p[x]$ , including the modular images  $f_p(x)$  and  $g_p(x)$ . Since the notation  $\mathbf{gcd}(f_p(x), g_p(x))$  is going to be used repeatedly, for brevity denote by  $e_p(x)$  the **gcd** calculated by Euclidean algorithm for  $f_p(x), g_p(x)$ . Let us stress that  $\mathbf{gcd}(f_p(x), g_p(x))$  is not determined uniquely, since for any non-zero  $t \in \mathbb{Z}_p$  the product  $t \cdot \mathbf{gcd}(f_p(x), g_p(x))$  also is a **gcd** for  $f_p(x), g_p(x)$ . We are denoting just *one* of these **gcd**'s (namely, that computed by the Euclidean algorithm) by  $e_p(x)$ . This  $e_p(x)$  is unique, since at each step of the Euclidean algorithm we have a unique action to take (to see this just consider the steps of "long division" used to divide  $f_p(x)$  by  $g_p(x)$  on field  $\mathbb{Z}_p$ ).

The main idea of the algorithm is to calculate the  $e_p(x) \approx \mathbf{gcd}(f_p(x), g_p(x))$  for some suitable  $p$ , and to reconstruct  $d(x) = \mathbf{gcd}(f(x), g(x))$  by it. We separate the process to four main problems that may occur, and show how to overcome each one to arrive to a correctly working algorithm.

##### 4.1. Problem 1. Avoiding the eliminating coefficients.

After reduction  $\varphi_p$  some of the coefficients of  $f(x)$  and  $g(x)$  may change or even eliminate. So their images  $f_p(x) = \varphi_p(f(x))$  and  $g_p(x) = \varphi_p(g(x))$  may keep very little information to reconstruct the  $d(x)$  based on  $e_p(x)$ .

EXAMPLE 4.1. If  $f(x) = 7x^2 + 22$  and  $g(x) = 49x^3 + 154x$  then for  $p = 7$  we get  $f_p(x) = 1$  and  $g_p(x) = 0$ . So these values contain no reliable information to reconstruct the  $\mathbf{gcd}(f(x), g(x))$ .

The first simple idea to avoid such eliminations is to take  $p$  larger than the absolute value of all coefficients of  $f(x)$  and  $g(x)$ . This, however, is not enough since a divisor  $h(x)$  of a polynomial  $f(x)$  may have coefficients, larger than those of  $f(x)$ . Moreover, using the cyclotomic polynomials for large enough  $n$ :

$$\phi_n(x) = \prod_{k=1, \dots, n; (k, n)=1} (x - e^{\frac{2i\pi k}{n}})$$

one can get divisors of  $f(x) = x^n - 1$  which have a coefficient larger than any pre-given number [5, 15, 11]. Since we do not know the divisors of  $f(x)$  and  $g(x)$ , we cannot be sure if the above mentioned large  $p$  will be large enough to prevent eliminations of coefficients of  $h(x)$ . To overcome this one can use the Landau-Mignotte bounds<sup>1</sup>, as it is done in [4, 15, 14]. For a polynomial  $f(x)$  given by (10) denote its norm by  $\|f(x)\| = \sqrt{\sum_{i=0}^n a_i^2}$ .

---

<sup>1</sup>In different sources the bounds on coefficients of the divisors are called differently, associating them by names of L. Landau or M. Mignotte or by both of them. These authors have different roles in development of the formulas, which in turn are consequence of a formula by A. L. Cauchy.



**THEOREM 4.2** (L. Landau, M. Mignotte). *Let  $f(x) = a_0x^n + \dots + a_n$  and  $h(x) = c_0x^k + \dots + c_k$  be non-zero polynomials in  $\mathbb{Z}[x]$ . If  $h(x)$  is a divisor of  $f(x)$ , then:*

$$(11) \quad \sum_{i=0}^n |c_i| \leq 2^k \cdot |c_0/a_0| \cdot \|f(x)\|.$$

The proof is based on calculations on complex numbers, and it can be found, for example, in [15, 11]. We are going to use the Landau-Mignotte bounds in the following two shapes:

**COROLLARY 4.3.** *In notations of Theorem 4.2 there is the following upper bound for the coefficients of  $h(x)$ :*

$$(12) \quad |c_i| \leq N_f = 2^{n-1} \|f(x)\|.$$

*Proof.* To obtain this from (11) first notice that  $|c_0/a_0| \leq 1$ .

Next, if  $k = \deg h(x) = \deg f(x) = n$ , then  $f(x) = r \cdot h(x)$ , where  $r$  is a non-zero integer. Then  $|c_i| \leq \max\{|c_i| \mid i = 0, \dots, n\} \leq \max\{|a_i| \mid i = 0, \dots, n\} \leq \|f(x)\|$ .

Finally, if  $k = \deg h(x) \leq n - 1$  ( $k$  is unknown to us), then we can simply replace in (11) the value  $2^k$  by  $2^{n-1}$ .  $\square$

**REMARK 4.4.** In literature they use the rather less accurate bound  $|c_i| \leq 2^n \|f(x)\|$ , but the second paragraph of our proof above allows to replace  $2^n$  by  $2^{n-1}$ . See also Remark 6.7.

**COROLLARY 4.5.** *In notations of Theorem 4.2, if  $h(x)$  also is a divisor of the polynomial  $g(x) = b_0x^m + \dots + b_m$ , then there is the following upper bound for the coefficients of  $h(x)$ :*

$$(13) \quad |c_i| \leq N_{f,g} = 2^{\min\{n,m\}} \cdot \mathbf{gcd}(a_0, b_0) \cdot \min \left\{ \frac{\|f(x)\|}{|a_0|}, \frac{\|g(x)\|}{|b_0|} \right\}.$$

*Proof.* To obtain this from (11) just notice that if  $h(x)$  is a common divisor for  $f(x)$  and  $g(x)$ , then its leading coefficient  $c_0$  divides both  $a_0$  and  $b_0$ .  $\square$

Formula (13) provides the hint to overcome Problem 1 about eliminating coefficients, mentioned at the start of this subsection. Although the divisors  $h(x)$  of  $f(x)$  and  $g(x)$  are yet unknown, we can compute  $N_{f,g}$  and take  $p > N_{f,g}$ . If we apply the reduction  $\varphi_p$  for this  $p$  we can be sure that none of the coefficients of  $h(x)$  has changed “much” under that homomorphism, for,  $\varphi_p$  does not alter the non-negative coefficients of  $h(x)$ , and it just adds  $p$  to all negative coefficients of  $h(x)$ . The same holds true for  $d(x) = \mathbf{gcd}(f(x), g(x))$ .

#### 4.2. Problem 2. Negative coefficients and reconstruction of the pre-image.

The reduction  $\varphi_p$  is not a bijection, and  $d_p(x)$  has infinitely many pre-images in  $\mathbb{Z}[x]$ . But the relatively uncomplicated relationship between coefficients of  $d(x)$  and  $d_p(x)$ , obtained in previous subsection, may allow us to reconstruct  $d(x)$  if we know  $d_p(x)$ . The condition  $p > N_{f,g}$  puts a restriction on the pre-image  $d(x)$ : the coefficients of  $d(x)$  are either equal to respective coefficients of  $d_p(x)$  (if they are non-negative), or they

are the respective coefficients of  $d_p(x)$  minus  $p$  (if they are negative). Reconstruction may cause problems connected with negative coefficients.

EXAMPLE 4.6. If for some polynomials  $f(x), g(x)$  we have  $N_{f,g} = 15$ , we can take the prime, say,  $p = 17 > N_{f,g}$ . Assume we have somehow calculated  $d_{17}(x) = 12x^3 + 3x + 10$ , we can be sure that  $d(x)$  is not the pre-image  $29x^3 - 17x^2 + 20x + 27$  because  $d(x)$  cannot have coefficients greater than 15 by absolute value. But we still cannot be sure if the pre-image  $d(x)$  is  $12x^3 + 3x + 10$ , or  $-5x^3 + 3x + 10$ , or maybe  $-5x^3 - 14x - 7$ .

It is easy to overcome this by just taking a larger value:

$$p > 2 \cdot N_{f,g}.$$

If the coefficient  $c_i$  of  $d(x)$  is non-negative, then  $\varphi_p(c_i) = c_i < p/2$ , and if it is negative, then  $\varphi_p(c_i) = c_i + p > p/2$ . This provides us with the following very simple algorithm to reconstruct  $d(x)$  if we have already computed  $d_p(x)$  for sufficiently large prime  $p$ .

ALGORITHM 4.1 (The polynomial reconstruction by modular image).

*Input:* For an unknown polynomial  $d(x) \in \mathbb{Z}[x]$  we know the upper bound  $N$  of absolute values of its coefficients, and for arbitrarily large prime number  $p$  we have the modular image  $d_p(x) = b_{0,p}x^k + \dots + b_{k,p} \in \mathbb{Z}_p[x]$ .

*Reconstruct the polynomial  $d(x)$ .*

01. Choose any prime  $p > 2 \cdot N$ .
02. Set  $k = \deg d_p(x)$ .
03. Set  $i = 0$ .
04. While  $i \leq k$
05.   if  $b_{i,p} < p/2$
06.       set  $b_i = b_{i,p}$ ;
07.   else
08.       set  $b_i = b_{i,p} - p$ ;
10.   set  $i = i + 1$ .
11. Output  $d(x) = b_0x^k + \dots + b_k$ .

#### 4.3. Problem 3. Finding the correct fold of the modular gcd of right degree.

Now additionally assume the polynomials  $f(x), g(x) \in \mathbb{Z}[x]$  to be primitive. Since  $\text{cont}(f(x))$  and  $\text{cont}(g(x))$  are defined up to the sign  $\pm 1$ , we can without loss of generality admit the leading coefficients of  $f(x), g(x)$  to be positive.

Below, in Problem 4, we will see that for some  $p$  the polynomial  $e_p(x)$ , computed by the Euclidean algorithm in  $\mathbb{Z}_p[x]$ , may not be the image of  $d(x)$  and, moreover, its degree may be different from that of  $d(x)$ . This means that by applying Algorithm 4.1 to  $e_p(x)$  we may not obtain  $d(x)$ . Assume, however, we have a  $p$ , which meets the condition  $p > 2 \cdot N_{f,g}$  and for which:

$$(14) \quad \deg d(x) = \deg e_p(x).$$

By Corollary 2.3 a common divisor of  $f(x), g(x)$  is the  $d(x) = \mathbf{gcd}(f(x), g(x))$  if and only if it is primitive and if its degree is the maximum of degrees of all common divisors. Since  $\varphi_p$  does not change the degree of  $d(x)$ , we get by Lemma 2.4 (applied

for the field  $K = \mathbb{Z}_p$  that  $d_p(x)$  is a **gcd** of  $f_p(x), g_p(x)$  in  $\mathbb{Z}_p[x]$ . This correspondence surely is not on-to-one, because in  $\mathbb{Z}[x]$  the **gcd** is calculated up to the unit element of  $\mathbb{Z}[x]$ , which is  $\pm 1$ , whereas in  $\mathbb{Z}_p[x]$  the **gcd** is calculated up to the unit element of  $\mathbb{Z}_p[x]$ , which can be any non-zero number  $t \in \mathbb{Z}_p^* = \{1, \dots, p-1\}$ . So the polynomial  $e_p(x)$  calculated by the Euclidean algorithm may not be the image  $d_p(x)$  of  $d(x)$ .

EXAMPLE 4.7. For  $f(x) = x^2 + 4x + 3$  and  $g(x) = x^2 + 2x + 1$  whichever prime  $p > 4$  we take, we will get by the Euclidean algorithm:

$$e_p(x) = \mathbf{gcd}(f_p(x), g_p(x)) = 2x + 2 \in \mathbb{Z}_p[x].$$

But in  $\mathbb{Z}[x]$  we have  $d(x) = x + 1$ . So regardless how large  $p$  we choose, we will never get  $\varphi_p(x + 1) = 2x + 2$ .

In other words, we are aware that the image  $d_p(x)$  is one of the folds  $t \cdot e_p(x)$  of  $e_p(x)$  for some  $t \in \{1, \dots, p-1\}$ , but we are not aware which  $t$  is that.

The leading coefficient  $c_0 = \text{lc}(d(x))$  of  $d(x)$  can also be assumed to be positive. Denote by  $w$  the positive **gcd**( $a_0, b_0$ ). Since both  $c_0$  and  $w$  are not altered by  $\varphi_p$ , their fraction  $w/c_0$  also is not altered. Take such a  $t$  that:

$$(15) \quad \text{lc}(t \cdot e_p(x)) = w.$$

Even if  $t \cdot e_p(x)$  is not the image  $d_p(x)$ , it is the image of  $l \cdot d(x)$ , where  $l$  divides  $w/c_0$ . If we calculate the pre-image  $k(x) \in \mathbb{Z}[x]$  of  $t \cdot e_p(x)$  by Algorithm 4.1, we will get a polynomial, which is either  $d(x)$  or is some fold of  $d(x)$ . Since  $f(x), g(x)$  are primitive, it remains to go to the primitive part  $d(x) = \text{pp}(k(x))$ .

The general case, when  $f(x), g(x)$  may not be primitive, can easily be reduced to this: for arbitrary  $f(x), g(x)$  take their decompositions by formula (4) and set

$$(16) \quad r = \mathbf{gcd}(\text{cont}(f(x)), \text{cont}(g(x))) \in \mathbb{Z}.$$

Then assign  $f(x) = \text{pp}(f(x))$ ,  $g(x) = \text{pp}(g(x))$  and do the steps above for these new polynomials. After the  $d(x) = \text{pp}(k(x))$  is computed, we get the final answer as  $r \cdot d(x) = r \cdot \text{pp}(k(x))$

Notice that for Algorithm 4.1 we need  $p$  to be greater than any coefficient  $|c_i|$  of the polynomial we reconstruct. The bound  $p > 2 \cdot N_{f,g}$  assures that  $p$  meets this condition for  $d(x)$ . We, however, reconstruct not  $d(x)$  but  $l \cdot d(x)$ , which may have larger coefficients. One could overcome this point by taking  $p > w \cdot 2 \cdot N_{f,g}$  but this is not necessary because, as we see later, while the Big prime modular **gcd** algorithm works, the value of  $p$  will grow and this issue will be covered.

#### 4.4. Problem 4. Finding the right degree for the modular gcd.

As we saw, one can reconstruct  $d(x)$  if we find a  $p > 2 \cdot N_{f,g}$  such that the condition (14) holds. Consider an example to see that (14) may actually not hold for some  $p$  even if  $\varphi_p$  is not altering the coefficients of  $f(x)$  and  $g(x)$ !

EXAMPLE 4.8. For  $f(x) = x^2 + 1$  and  $g(x) = x + 1$  we have  $d(x) = \mathbf{gcd}(f(x), g(x)) = 1$ . Taking  $p = 2$  we get  $f_2(x) = x^2 + 1$  and  $g_2(x) = x + 1$ . In  $\mathbb{Z}_2[x]$  we have  $f_2(x) = x^2 + 1 = x^2 + 1^2 = (x + 1)(x + 1)$ , thus,  $e_2(x) = \mathbf{gcd}(f_2(x), g_2(x)) = x + 1$ . We get that  $1 = \deg(x + 1) > \deg(x) = 0$ . In particular, whatever  $t$  we take,  $t \cdot (x + 1)$  is not the image of  $d(x) = 1$  under  $\varphi_2$ .

The idea to overcome this problem is to show that the number of primes  $p$ , for which (14) falsifies, is “small”. So if the selected  $p$  is not suitable, we take another  $p$  and do the calculation again by the new prime. And we will not have to repeat these steps for many times (we will turn to this point in Section 6).

The proof of the following theorem and the definition of the resultant  $\text{res}(f(x), g(x))$  (that is, of the determinant of the Sylvester matrix  $S_{f,g}$  of polynomials  $f(x), g(x)$ ) can be found, for example, in [10, 15, 8, 11]. The resultant is a comfortable tool to detect if the given polynomials are coprime:

**THEOREM 4.9.** *Let  $R$  be an integrity domain. The polynomials  $f(x), g(x) \in R[x]$  are coprime if and only if  $\text{res}(f(x), g(x)) \neq 0$ .*

The following fact in a little different shape can be found in [15] or [4]:

**COROLLARY 4.10.** *If the prime  $p$  does not divide at least one of the leading coefficients  $a_0, b_0$  of polynomials, respectively,  $f(x), g(x) \in \mathbb{Z}[x]$  then  $\deg d(x) \leq \deg e_p(x)$ . If  $p$  also does not divide  $R = \text{res}(f(x)/d(x), g(x)/d(x))$ , where  $d(x) = \mathbf{gcd}(f(x), g(x))$ , then*

$$(17) \quad \deg d(x) = \deg d_p(x) = \deg \mathbf{gcd}(f_p(x), g_p(x)) = \deg e_p(x).$$

*Proof.* Since  $c_0 = \text{lc}(d(x))$  divides  $w = \mathbf{gcd}(a_0, b_0)$ , then  $\varphi_p(c_0) \neq 0$  by the choice of  $p$ . Thus,  $\deg d(x) = \deg d_p(x) \leq \deg \mathbf{gcd}(f_p(x), g_p(x))$ .

Since  $d_p(x) \neq 0$ , we can consider the fractions  $f_p(x)/d_p(x)$  and  $g_p(x)/d_p(x)$  in  $\mathbb{Z}_p[x]$ . From unique factorizations of  $f_p(x)$  and  $g_p(x)$  in UFD  $\mathbb{Z}_p[x]$  it is very easy to deduce that

$$e_p(x) \approx \mathbf{gcd}(f_p(x), g_p(x)) \approx d_p(x) \cdot \mathbf{gcd}(f_p(x)/d_p(x), g_p(x)/d_p(x)).$$

In particular,  $\deg d(x) = \deg d_p(x) \leq \deg e_p(x)$ . And the inequality  $\deg d_p(x) \neq \deg e_p(x)$  may occur only if

$$\deg \mathbf{gcd}(f_p(x)/d_p(x), g_p(x)/d_p(x)) > 0,$$

that is, when  $f_p(x)/d_p(x)$  and  $g_p(x)/d_p(x)$  are not coprime in  $\mathbb{Z}_p[x]$  or, by Theorem 4.9, when  $\text{res}(f_p(x)/d_p(x), g_p(x)/d_p(x)) = 0$ . The latter is the determinant of Sylvester matrix  $S_{f_p/d_p, g_p/d_p}$ . Consider the matrix rings homomorphism (matrix modular reduction)

$$\varphi_p : M_{m+n}(\mathbb{Z}) \rightarrow M_{m+n}(\mathbb{Z}_p),$$

where  $n = \deg f(x)$ ,  $m = \deg g(x)$  (as mentioned earlier we use the same symbol  $\varphi_p$  for numeric, polynomial and matrix reductions). Since,  $\varphi_p(S_{f/d, g/d}) = S_{f_p/d_p, g_p/d_p}$ , and since the determinant of a matrix is a sum of products of its elements, we get

$$R_p = \varphi_p(R) = \varphi_p(\text{res}(f(x)/d(x), g(x)/d(x))) = \text{res}(f_p(x)/d_p(x), g_p(x)/d_p(x)).$$

So  $R_p$  can be zero if and only if  $R$  is divisible by  $p$ . The polynomials  $f(x)/d(x)$  and  $g(x)/d(x)$  are coprime in  $\mathbb{Z}[x]$  and their resultant is not zero by Theorem 4.9. And  $R$  cannot be a positive integer divisible by  $p$  since that contradicts the condition of this corollary.  $\square$

Corollary 4.10 shows that if for some  $p$  the equality (14) does not hold for polynomials  $f(x), g(x) \in \mathbb{Z}[x]$ , then  $p$  divides either  $a_0$  and  $b_0$ , or it divides the resultant

$R$ . We do not know  $R$ , since we do not yet know  $d(x)$  to calculate the resultant  $R = \text{res}(f(x)/d(x), g(x)/d(x))$ . But, since the number of such primes  $p$  is just finite, we can arrive to the right  $p$  after trying the process for a few primes. We will turn to this again in Section 6.

## 5. THE BIG PRIME MODULAR **gcd** ALGORITHM

Four steps of the previous section provide us with the following procedure. We keep all the notations from Section 4. Take the primitive polynomials  $f(x), g(x) \in \mathbb{Z}[x]$ . Without loss of generality we may assume  $a_0, b_0 > 0$ . Take any  $p > 2 \cdot N_{f,g}$ . Then  $p \nmid w = \mathbf{gcd}(a_0, b_0)$ , since  $a_0, b_0 \leq N_{f,g}$ . Calculate  $e_p(x) = \mathbf{gcd}(f_p(x), g_p(x))$  in  $\mathbb{Z}_p[x]$  by Euclidean algorithm. Then choose  $t$  so that (15) holds. Construct  $k(x)$  applying Algorithm 4.1 to  $t \cdot e_p(x)$ . If the primitive part  $d(x) = \text{pp}(k(x))$  divides both  $f(x)$  and  $g(x)$ , then the **gcd** for these primitive polynomials is found:  $d(x) = \mathbf{gcd}(f(x), g(x))$ . That follows from consideration about divisor degrees above: if  $f(x), g(x)$  had a common divisor  $h(x)$  of degree greater than  $\deg d(x)$ , then, since the degree of  $h(x)$  is not altered by  $\varphi_p$ , we would get  $\deg h_p(x) > \deg d_p(x) = \deg d(x) = \deg e_p(x)$ , which contradicts the maximality of  $\deg d_p(x)$  by Lemma 2.4.

This means that if for  $p > 2 \cdot N_{f,g}$  we get  $d(x) \nmid f(x)$  or  $d(x) \nmid g(x)$ , we have the case when  $p$  divides the resultant  $R$ . Then we just ignore the calculated polynomial, choose another  $p > 2 \cdot N_{f,g}$  and redo the steps for it. Repeating these steps for finitely many times, we will eventually arrive to the correct  $d(x)$  for the primitive polynomials  $f(x), g(x)$ .

The case of arbitrary non-zero polynomials can easily be reduced to this. By arguments mentioned earlier: we should calculate  $d(x)$  for primitive polynomials  $\text{pp}(f(x))$  and  $\text{pp}(g(x))$ , and then output the final answer as  $r \cdot d(x)$ , where  $r$  is defined by (16). The process we described is the traditional form of the Big prime modular **gcd** algorithm.

**REMARK 5.1.** Since our approach in Section 4 evolved the maximality of degrees of the common divisors, we can shorten some of the steps of our algorithm. Let us store in a variable, say,  $D$  the minimal value for which we already know it is not the  $\deg \mathbf{gcd}(f(x), g(x))$ . As an initial  $D$  we may take, say,  $D = \min\{\deg f(x), \deg g(x)\} + 1$ . Each time we calculate  $e_p(x) = \mathbf{gcd}(f_p(x), g_p(x))$ , check if  $\deg e_p(x)$  is equal to or larger than the current  $D$ . If yes, we already know that we have an “inappropriate”  $p$ . Then we no longer need use Algorithm 4.1 to reconstruct  $k(x)$  and to get  $d(x) = \text{pp}(k(x))$ . We just skip these steps and proceed to the next  $p$ . Reconstruct  $d(x)$  and check if  $d(x) \mid f(x)$  and  $d(x) \mid g(x)$  *only* when  $\deg e_p(x) < D$ . Then, if  $d(x)$  does not divide  $f(x)$  or  $g(x)$ , we have discovered a new bound  $D$  for  $\deg(\mathbf{gcd}(f(x), g(x)))$ . So set  $D = \deg e_p(x)$  and proceed to the next  $p$ . If in next step we get  $\deg e_p(x) \geq D$ , we will again be aware that the steps of reconstruction of  $d(x)$  need be skipped.

We constructed the following algorithm:

**ALGORITHM 5.1** (Big prime modular **gcd** algorithm).

*Input:* non-zero polynomials  $f(x), g(x) \in \mathbb{Z}[x]$ .

*Calculate their greatest common divisor*  $\mathbf{gcd}(f(x), g(x)) \in \mathbb{Z}[x]$ .

01. Calculate  $\text{cont}(f(x))$ ,  $\text{cont}(g(x))$  in Euclidean domain  $\mathbb{Z}$ , choose their signs so that  $\text{sgn cont}(f(x)) = \text{sgn lc}(f(x))$  and  $\text{sgn cont}(g(x)) = \text{sgn lc}(g(x))$ .
02. Set  $f(x) = \text{pp}(f(x))$  and  $g(x) = \text{pp}(g(x))$ .
03. Calculate  $r$  in Euclidean domain  $\mathbb{Z}$  by (16).
04. Set  $a_0 = \text{lc}(f(x))$  and  $b_0 = \text{lc}(g(x))$  (they are positive by our selection of signs for  $\text{cont}(f(x))$  and  $\text{cont}(g(x))$ ).
05. Calculate the positive  $w = \mathbf{gcd}(a_0, b_0)$  in Euclidean domain  $\mathbb{Z}$ .
06. Set  $D = \min\{\deg f(x), \deg g(x)\} + 1$ .
07. Compute the Landau-Mignotte bound  $N_{f,g}$  by (13).
08. Choose a new prime number  $p > 2 \cdot N_{f,g}$ .
09. Apply the reduction  $\varphi_p$  to calculate the modular images  $f_p(x), g_p(x) \in \mathbb{Z}_p[x]$ .
10. Calculate  $e_p(x) = \mathbf{gcd}(f_p(x), g_p(x))$  in the Euclidean domain  $\mathbb{Z}_p[x]$ .
11. If  $D \leq \deg e_p(x)$
12.     go to step 08;
13. else
14.     choose a  $t$  such that the  $\text{lc}(t \cdot e_p(x)) = w$ ;
15.     call Algorithm 4.1 to calculate the preimage  $k(x)$  of  $t \cdot e_p(x)$ ;
16.     calculate  $\text{cont}(k(x))$  in Euclidean domain  $\mathbb{Z}$ ;
17.     set  $d(x) = \text{pp}(k(x)) = k(x)/\text{cont}(k(x))$ ;
18.     if  $d(x)|f(x)$  and  $d(x)|g(x)$
19.         go to step 23;
20.     else
21.         set  $D = \deg e_p(x)$ ;
22.         go to step 08.
23. Output the result:  $\mathbf{gcd}(f(x), g(x)) = r \cdot d(x)$ .

Turning back to Remark 5.1, notice that for some prime numbers  $p$  we skip the steps 14 – 18 of Algorithm 5.1, and directly jump to the step 08. In fact, Remark 5.1 has mainly theoretical purpose to display how usage of UFD properties and comparison of divisor degrees may reduce some of the steps of the Big prime modular  $\mathbf{gcd}$  algorithm. In practical examples the set of primes we use contains few primes dividing  $R = \text{res}(f(x)/d(x), g(x)/d(x))$ , so we may not frequently get examples where the steps 14 – 18 are skipped.

EXAMPLE 5.2. Let us apply Algorithm 5.1 to polynomials (1) mentioned in Knuth's example above. Since  $\|f(x)\| = \sqrt{113}$  and  $\|g(x)\| = \sqrt{570}$ ,

$$N_{f,g} = 2^{\min\{8,6\}} \cdot \mathbf{gcd}(1,3) \cdot \min \left\{ \frac{\sqrt{113}}{1}, \frac{\sqrt{570}}{3} \right\} < 512.$$

And we can take the prime  $p = 1031 > 2 \cdot N_{f,g}$ . It is not hard to compute that  $\mathbf{gcd}(f_{1031}(x), g_{1031}(x)) \approx 1$ . So  $f(x)$  and  $g(x)$  are coprime. It is worth to compare  $p = 1031$  with much smaller values  $p = 67$  and  $p = 37$  obtained below for the same polynomials (1) in Example 8.1 using the modified Algorithm 8.1.

In [11] we also apply Algorithm 5.1 to other polynomials with cases when the polynomials are not coprime.

## 6. ESTIMATING THE PRIME DIVISORS OF THE RESULTANT

Although at the start of the Big prime modular **gcd** algorithm we cannot compute the resultant  $R = \text{res}(f(x)/d(x), g(x)/d(x))$  for the given  $f(x), g(x) \in \mathbb{Z}[x]$  (we do not know  $d(x)$ ), we can nevertheless estimate the value of  $R$  and the number of its prime divisors. Denote:

$$(18) \quad \begin{aligned} A_{f,g} &= \sqrt{(n+1)^m(m+1)^n \cdot N_f^m N_g^n} \\ &= 2^{2nm-n-m} \sqrt{(n+1)^m(m+1)^n} \cdot \|f(x)\|^m \|g(x)\|^n. \end{aligned}$$

LEMMA 6.1. *For any polynomials  $f(x), g(x) \in \mathbb{Z}[x]$  and for any of their common divisors  $d(x)$  the following holds:*

$$|\text{res}(f(x)/d(x), g(x)/d(x))| = |S_{f/d, g/d}| \leq A_{f,g}.$$

*Proof.* By Corollary 4.3 the coefficients of fractions  $f(x)/d(x)$  and  $g(x)/d(x)$  are bounded, respectively, by  $N_f = 2^{n-1}\|f(x)\|$  and  $N_g = 2^{m-1}\|g(x)\|$ , where  $n = \deg f(x)$ ,  $m = \deg g(x)$ . Since the numbers of summands in these fractions are at most  $n+1$  and  $m+1$ , respectively, we get:

$$\|f(x)/d(x)\| \leq \sqrt{(n+1)N_f^2} \quad \text{and} \quad \|g(x)/d(x)\| \leq \sqrt{(m+1)N_g^2}.$$

Applying the Hadamard's maximal determinant bound [15] to the Sylvester matrix  $S_{f/d, g/d}$ , we get that

$$|R| = |S_{f/d, g/d}| \leq \left(\sqrt{(n+1)N_f^2}\right)^m \cdot \left(\sqrt{(m+1)N_g^2}\right)^n.$$

□

The bound of (18) is very rough. To see this apply it to the polynomials (1) of Knuth's example:

EXAMPLE 6.2. For polynomials (1) we have  $\|f(x)\| = \sqrt{113}$  and  $\|g(x)\| = \sqrt{570}$ . So we can estimate  $N_f < 1408$ ,  $N_g < 768$  and  $N_{f,g} < 512$ . Thus:

$$|R| \leq \sqrt{(8+1)^6(6+1)^8} \cdot 1408^6 \cdot 570^8 = \omega = 1.6505374299582118582810249858265e+48,$$

which is a too large number to comfortably operate with.

REMARK 6.3. If in Algorithm 5.1 we use a prime

$$(19) \quad p > 2 \cdot A_{f,g},$$

then we will get that  $p \nmid R = \text{res}(f(x)/d(x), g(x)/d(x))$  whatever the greatest common divisor  $d(x)$  be. And, clearly,  $p \nmid w$  holds for  $w = \mathbf{gcd}(a_0, b_0)$ . So in this case Algorithm 5.1 will output the correct  $\text{pp}(k(x))$  using just one  $p$ , and we will not have to take another  $p \nmid w$  after step 18. However, Example 6.2 shows why it is *not* reasonable to chose  $p$  by the rule (19) to have in Algorithm 5.1 one cycle only: it is easier to go via a few cycles for smaller  $p$ 's rather than to operate with a huge  $p$ , which is two times larger than the bound  $\omega$  obtained in Example 6.2.

Nevertheless, the bound  $A_{f,g}$  may be useful if we remember that the process in Algorithm 5.1 concerned not the value of  $\text{res}(f(x)/d(x), g(x)/d(x))$  but the *number of its distinct prime divisors*. Let us denote by  $p_k\#$  the product of the first  $k$  primes:  $p_k\# = p_1 \cdot p_2 \cdots p_k$  (where  $p_1 = 2$ ,  $p_2 = 3$ , etc.). They sometimes call  $p_k\#$  the “ $k$ ’th primorial”. The following is essential:

LEMMA 6.4. *The number of pairwise distinct prime divisors of a positive integer  $n$  is less or equal to  $\max \{k \mid p_k\# \leq n\}$ .*

From Lemma 6.1 and Lemma 6.4 easily follows:

COROLLARY 6.5. *For any polynomials  $f(x), g(x) \in \mathbb{Z}[x]$  and for any of their common divisors  $d(x)$  the number of pairwise distinct prime divisors of  $\text{res}(f(x)/d(x), g(x)/d(x))$  is at most  $k$ , where  $k$  is the largest number for which  $p_k\# \leq A_{f,g}$ .*

Primorial (as a function on  $k$ ) grows very rapidly. Say, for  $k = 10$  it is more than six billions:  $p_{10}\# = 6,469,693,230$ . This observation allows to use the bound  $A_{f,g}$  in the following way: although the value of  $A_{f,g}$  as a function on  $n = \deg f(x)$ ,  $m = \deg g(x)$  and on the coefficients of  $f(x)$  and  $g(x)$  grows rapidly, the number of its distinct prime divisors, may not be “very large” thanks to the fact that  $p_k\#$  also grows rapidly. Consider this on polynomials and values from Example 6.2:

EXAMPLE 6.6. It is easy to compute that:

$$p_{30}\# = 3.1610054640417607788145206291544e + 46 < \omega$$

and

$$p_{31}\# = 4.014476939333036189094441199026e + 48 > \omega,$$

where  $\omega$  is the large number from Example 6.2. This means that the number of prime divisors of  $R = \text{res}(f(x)/d(x), g(x)/d(x))$ , whatever the divisor  $d(x)$  be, is not greater than 30. And whichever  $30 + 1 = 31$  distinct primes we take, at least one of them will *not* be a divisor of  $R$ . That is, Algorithm 5.1 for the polynomials of Knuth’s example will output the correct answer in not more than 31 cycles. We *cannot* find 31 primes  $p \nmid w$  so that Algorithm 5.1 arrives to a wrong  $d(x) = \text{pp}(k(x))$  on step 18 for all of them.

REMARK 6.7. Let us stress that estimates on the number of prime divisors of the resultant and the analog of Algorithm 7.1 can be found elsewhere, for example, in [15]. So the only news we have is that here we use a slightly better value for  $N_f$  and  $N_g$  to get  $2^{n+m}$  times smaller bound for  $A_{f,g}$ . Namely, in Corollary 4.3 we estimate  $|c_i|$  not by  $2^n \|f(x)\|$  but by  $2^{n-1} \|f(x)\|$  (see (12) and Remark 4.4). This makes the bound  $A_{f,g}$  in formula (18)  $2^{n+m}$  times lower, since  $N_f$  and  $N_g$  appear  $m$  and  $n$  times respectively.

## 7. AN ALGORITHM TO CHECK COPRIME POLYNOMIALS

The first application of the bounds found in previous section is an algorithm checking if the given polynomials  $f(x), g(x) \in \mathbb{Z}[x]$  are coprime. Present the polynomials as  $f(x) = \text{cont}(f(x)) \cdot \text{pp}(f(x))$  and  $g(x) = \text{cont}(g(x)) \cdot \text{pp}(g(x))$ . If  $r = \text{gcd}(\text{cont}(f(x)), \text{cont}(g(x))) \neq 1$ , then  $f(x), g(x)$  are not coprime, and we do not have to check the primitive parts, at all.



If  $r \approx 1$ , then switch to the polynomials  $f(x) = \text{pp}(f(x))$  and  $g(x) = \text{pp}(g(x))$ . By Corollary 6.5 the number of distinct prime divisors of  $\text{res}(f(x)/d(x), g(x)/d(x))$  is less or equal to  $k$ , where  $k$  is the largest number for which  $p_k\# \leq A_{f,g}$ .

Consider any  $k+1$  primes  $p_1, \dots, p_{k+1}$ , each not dividing  $w = \mathbf{gcd}(a_0, b_0)$ , where  $a_0 = \text{lc}(f(x))$  and  $b_0 = \text{lc}(g(x))$ . If  $\mathbf{gcd}(f_{p_i}(x), g_{p_i}(x)) = 1$  for at least one  $p_i$ , then  $f(x)$  and  $g(x)$  are coprime because  $0 = \deg \mathbf{gcd}(f_{p_i}(x), g_{p_i}(x)) \geq \deg \mathbf{gcd}(f(x), g(x))$  and  $r \approx 1$ .

And if  $\mathbf{gcd}(f_{p_i}(x), g_{p_i}(x)) \neq 1$  for all  $i = 1, \dots, k+1$ , then  $f_{p_i}(x)$  and  $g_{p_i}(x)$  are not coprime for at least one  $p_i$ , which is not dividing  $\text{res}(f(x)/d(x), g(x)/d(x))$ . This means that  $f(x)$  and  $g(x)$  are not coprime. We got the following algorithm:

**ALGORITHM 7.1** (Coprime polynomials detection modular algorithm).

*Input: non-zero polynomials  $f(x), g(x) \in \mathbb{Z}[x]$ .*

*Detect if  $f(x)$  and  $g(x)$  are coprime.*

01. Calculate  $\text{cont}(f(x))$ ,  $\text{cont}(g(x))$  in Euclidean domain  $\mathbb{Z}$ .
02. Calculate  $r$  in Euclidean domain  $\mathbb{Z}$  by (16).
03. If  $r \not\approx 1$
04.     output the result:  $f(x)$  and  $g(x)$  are not coprime and stop.
05. Set  $a_0 = \text{lc}(f(x))$  and  $b_0 = \text{lc}(g(x))$ .
06. Calculate  $w = \mathbf{gcd}(a_0, b_0)$  in Euclidean domain  $\mathbb{Z}$ .
07. Set  $f(x) = \text{pp}(f(x))$  and  $g(x) = \text{pp}(g(x))$ .
08. Compute the bound  $A_{f,g}$  for polynomials  $f(x), g(x)$  by (18).
09. Find the maximal  $k$  for which  $p_k\# \leq A_{f,g}$ .
10. Set  $i = 1$ .
11. While  $i \neq k+1$
12.     choose a new prime  $p \nmid w$ ;
13.     apply the reduction  $\varphi_p$  to calculate the modular images  $f_p(x), g_p(x) \in \mathbb{Z}_p[x]$ ;
14.     calculate  $e_p = \mathbf{gcd}(f_p(x), g_p(x))$  in Euclidean domain  $\mathbb{Z}_p[x]$ ;
15.     if  $\deg e_{q_i} = 0$
16.         output the result:  $f(x)$  and  $g(x)$  are coprime and stop.
17.     set  $i = i + 1$ .
18. If  $i < k+1$
19.     go to step 12.
20. else
21.     output the result:  $f(x)$  and  $g(x)$  are not coprime.

Two important advantages of this algorithm are that here we use much smaller primes  $p$  (we just require  $p \nmid w$ , not  $p > 2 \cdot N_{f,g}$ ), and in Algorithm 7.1, unlike in Algorithm 5.1, we never need to find  $t$ , to compute the preimage  $k(x)$  of  $t \cdot \mathbf{gcd}(f_p(x), g_p(x))$  and the primitive part  $\text{pp}(k(x))$ .

**REMARK 7.1.** As it is mentioned by Knuth in [7], in a probabilistic sense the polynomials are much more likely to be coprime than the integer numbers. So it is reasonable

to first test by Algorithm 7.1 if the given polynomials  $f(x), g(x)$  are coprime, and only after that apply Algorithm 5.1 to find their **gcd** in case if they are not coprime. See also Algorithm 8.2, where we combine both these approaches with a better bound for prime  $p$ .

EXAMPLE 7.2. Apply Algorithm 7.1 to polynomials (1) from Knuth's example. As we saw in Example 6.6,  $k = 30$ . For  $p = 2$  we get  $f_2(x) = x^8 + x^6 + x^4 + x^3 + 1$ ,  $g_2(x) = x^6 + x^4 + x + 1$ , which are not coprime, since  $\mathbf{gcd}(f_2(x), g_2(x)) = x^2 + x + 1 \neq 1$ . And for  $p = 3$  we get  $f_3(x) = x^8 + x^6 + 2x^2 + 2x + 1$ ,  $g_3(x) = 2x^4 + 2x^2$ , which are coprime. So  $\mathbf{gcd}(f(x), g(x)) = 1$ .

EXAMPLE 7.3. If  $f(x) = x^2 + 2x + 1$  and  $g(x) = x + 1$ . Then  $N_f = 2\sqrt{6}$ ,  $N_g = \sqrt{2}$  and  $A_{f,g} < 39$ . Since  $2 \cdot 3 \cdot 5 \cdot 7 = 210 > 39$ , we get that  $k = 3$ , and  $\mathbf{gcd}(f(x), g(x)) \neq 1$  if  $\mathbf{gcd}(f_p(x), g_p(x)) \neq 1$  for any *four* primes (not dividing  $w$ ). It is easy to check that  $\mathbf{gcd}(f_p(x), g_p(x)) \neq 1$  for  $p = 2, 3, 5, 7$ .

## 8. OTHER MODIFICATIONS OF ALGORITHMS

The bounds mentioned in Section 6 can be applied to obtain modifications of Algorithm 5.1. Let us outline four ideas, of which only the last two will be written down as algorithms.

For the non-zero polynomials  $f(x), g(x) \in \mathbb{Z}[x]$  let us again start by computing  $r = \mathbf{gcd}(\text{cont}(f(x)), \text{cont}(g(x)))$  and switching to the primitive parts  $f(x) = \text{pp}(f(x))$  and  $g(x) = \text{pp}(g(x))$ , assuming that their leading coefficients  $a_0$  and  $b_0$  are positive. Calculate  $N_f$ ,  $N_g$  by Corollary 4.3,  $N_{f,g}$  by Corollary 4.5 and  $A_{f,g}$  by (18). Find the maximal  $k$  for which  $p_k \# \leq A_{f,g}$ . Then take any  $k+1$  primes  $p_1, \dots, p_{k+1}$  each greater than  $2 \cdot N_{f,g}$ . We do not know  $d(x)$ , but we are aware that the number of prime divisors of  $R = \text{res}(f(x)/d(x), g(x)/d(x))$  is less than equal to  $k$ . So at least one of the primes  $p_1, \dots, p_{k+1}$  is not dividing  $R$ . To find it compute the degrees of  $e_{p_i}(x)$  for all  $i = 1, \dots, k+1$ . Take any  $p_i$ , for which  $\deg e_{p_i}(x)$  is the minimal (in case there are more than one  $p_i$ 's with this property, take one of them, preferably, the smallest of all).

By our construction,  $\deg e_{p_i}(x) = \deg \mathbf{gcd}(f(x), g(x))$  holds. So we can proceed to the next steps: choose a  $t$ , such that  $\text{lc}(t \cdot e_{p_i}(x)) = w = \mathbf{gcd}(a_0, b_0)$ ; then find by Algorithm 4.1 the pre-image  $k(x)$  of  $t \cdot e_{p_i}(x)$ ; then proceed to its primitive part  $d(x) = \text{pp}(k(x))$ ; and then output the final answer as  $r \cdot d(x)$ .

The advantage of this approach is that we do not have to go via the steps 14–18 of Algorithm 5.1 for more than one prime  $p$ . Also, we do not have to take care of the variable  $D$ . But the disadvantage is that we have to compute  $e_{p_i}(x)$  for large primes for  $k+1$  times (whereas in Algorithm 5.1 the correct answer could be discovered after consideration of fewer primes). Clearly, the disadvantage is a serious obstacle, since repetitions for  $k+1$  large primes consumes more labour than the steps 14–18 of Algorithm 5.1. So this is just a theoretical idea, not an approach for an effective algorithm.

The disadvantage can be reduced in the following way: in previous arguments, after we find  $p_k\#$  and  $k$ , select the prime numbers  $p_1, \dots, p_{k+1}$  each satisfying the condition  $p_i \nmid w$ . This is a much weaker condition than the condition  $p_i > 2 \cdot N_{f,g}$  used above, so we will surely get smaller primes. Take  $M$  to be the minimum of all degrees  $\deg e_{p_i}(x)$  for all  $i = 1, \dots, k+1$ . Since none of the primes  $p_1, \dots, p_{k+1}$  divides  $w$ , for any  $i = 1, \dots, k+1$  we have  $\deg e_{p_i}(x) \geq \deg \mathbf{gcd}(f(x), g(x))$ . On the other hand, since at least one of the primes  $p_1, \dots, p_{k+1}$  does not divide  $R$ , we know that for that  $p_i$  the degree of  $e_{p_i}(x)$  is equal to  $\deg \mathbf{gcd}(f(x), g(x))$ . Combining these we get that  $\deg \mathbf{gcd}(f(x), g(x)) = M$ . Since we know  $M$ , we can take a prime  $p > 2 \cdot N_{f,g}$ , compute the  $e_p(x)$  and check its degree: if  $\deg e_p(x) \neq M$ , then we have a wrong  $p$  (we no longer need go the steps 14–18 of Algorithm 5.1 to discover that). Then choose a new value for  $p$  and repeat the step. And if  $\deg e_p(x) = M$ , then we have the right  $p$ . We calculate  $t$ , the pre-image  $k(x)$  of  $t \cdot e_p(x)$ , then  $d(x) = \text{pp}(k(x))$ , and output the answer  $r \cdot d(x)$  (see Algorithm 8.2 for a better version of this idea).

The third modification, not depending on  $A_{f,g}$  can be constructed by estimating  $\deg \mathbf{gcd}(f(x), g(x)) = \deg d(x)$  by means of an auxiliary prime number  $q$ . By Landau-Mignotte Theorem 4.2, if  $h(x) = c_0x^k + \dots + c_k$  is any divisor of the polynomials  $f(x) = a_0x^n + \dots + a_n$  and  $g(x) = b_0x^m + \dots + b_m$  then  $|c_i| \leq 2^k |c_0/a_0| \|f(x)\|$  and  $|c_i| \leq 2^k |c_0/b_0| \|g(x)\|$ . Since  $|c_0/a_0|$  is bounded by  $|\mathbf{gcd}(a_0, b_0)/a_0|$  and  $|c_0/b_0|$  is bounded by  $|\mathbf{gcd}(a_0, b_0)/b_0|$ , we get the following analog of (13):

$$(20) \quad |c_i| \leq 2^k \cdot \mathbf{gcd}(a_0, b_0) \cdot \min \left\{ \frac{\|f(x)\|}{|a_0|}, \frac{\|g(x)\|}{|b_0|} \right\}.$$

Now assume  $q$  is a prime not dividing  $w$ , and denote  $s(q, f, g) = \deg \mathbf{gcd}(f_q(x), g_q(x))$ . By Corollary 4.10,  $\deg d(x) \leq s(q, f, g)$ . We get for the coefficients of  $d(x)$  the following bound :  $|c_i| \leq M_{q,f,g}$ , where

$$(21) \quad M_{q,f,g} = 2^{s(q,f,g)} \cdot \mathbf{gcd}(a_0, b_0) \cdot \min \left\{ \frac{\|f(x)\|}{|a_0|}, \frac{\|g(x)\|}{|b_0|} \right\}.$$

$M_{q,f,g}$  is a better bound for the coefficients of  $d(x)$  because  $2^{s(q,f,g)}$  may be considerably less than  $2^{\min\{n,m\}}$ .

We can improve Algorithm 5.1, if we preliminarily find  $s(q, f, g)$  by calculating the  $\mathbf{gcd}(f_q(x), g_q(x))$  for an “auxiliary” prime  $q \nmid w$ , and then chose the “main” prime  $p$  by the rule  $p > 2 \cdot M_{q,f,g}$  (instead of  $p > 2 \cdot N_{f,g}$ ). Observe that if  $p > 2 \cdot M_{q,f,g}$  then also  $q \nmid w = \mathbf{gcd}(a_0, b_0)$  because  $\min \left\{ \frac{\|f(x)\|}{|a_0|}, \frac{\|g(x)\|}{|b_0|} \right\} \geq 1$ . Additionally, we can introduce the variable  $D$  to store those values  $\deg \mathbf{gcd}(f_p(x), g_p(x))$  that we know are greater than  $\deg d(x)$ . We get the following algorithm:

**ALGORITHM 8.1** (Big prime modular  $\mathbf{gcd}$  algorithm with a preliminary estimate on divisor degree).

*Input:* non-zero polynomials  $f(x), g(x) \in \mathbb{Z}[x]$ .

*Calculate their greatest common divisor*  $\mathbf{gcd}(f(x), g(x)) \in \mathbb{Z}[x]$ .

01. Calculate  $\text{cont}(f(x))$ ,  $\text{cont}(g(x))$  in Euclidean domain  $\mathbb{Z}$ , choose their signs so that  $\text{sgn cont}(f(x)) = \text{sgn lc}(f(x))$  and  $\text{sgn cont}(g(x)) = \text{sgn lc}(g(x))$ .
02. Set  $f(x) = \text{pp}(f(x))$  and  $g(x) = \text{pp}(g(x))$ .

03. Calculate  $r$  in Euclidean domain  $\mathbb{Z}$  by (16).
04. Set  $a_0 = \text{lc}(f(x))$  and  $b_0 = \text{lc}(g(x))$  (they are positive by our selection of signs for  $\text{cont}(f(x))$  and  $\text{cont}(g(x))$ ).
05. Calculate the positive  $w = \mathbf{gcd}(a_0, b_0)$  in Euclidean domain  $\mathbb{Z}$ .
06. Set  $D = \min\{\deg f(x), \deg g(x)\} + 1$ .
07. Choose a prime number  $q \nmid w$ .
08. Apply the reduction  $\varphi_q$  to calculate the modular images  $f_q(x), g_q(x) \in \mathbb{Z}_q[x]$ .
09. Calculate  $e_q(x) = \mathbf{gcd}(f_q(x), g_q(x))$  in the Euclidean domain  $\mathbb{Z}_q[x]$ .
10. Set  $s(q, f, g) = \deg e_q(x)$ .
11. Calculate  $M_{q,f,g}$  by (21) using the value of  $s(q, f, g)$ .
12. Choose a new prime number  $p > 2 \cdot M_{q,f,g}$ .
13. Apply the reduction  $\varphi_p$  to calculate the modular images  $f_p(x), g_p(x) \in \mathbb{Z}_p[x]$ .
14. Calculate  $e_p(x) = \mathbf{gcd}(f_p(x), g_p(x))$  in Euclidean domain  $\mathbb{Z}_p[x]$ .
15. If  $D \leq \deg e_p(x)$
16.     go to step 12.
17. else
18.     choose a  $t$  such that the  $\text{lc}(t \cdot e_p(x)) = w$ ;
19.     call Algorithm 4.1 to calculate the preimage  $k(x)$  of  $t \cdot e_p(x)$ ;
20.     calculate  $\text{cont}(k(x))$  in Euclidean domain  $\mathbb{Z}$ ;
21.     set  $d(x) = \text{pp}(k(x)) = k(x)/\text{cont}(k(x))$ ;
22.         if  $d(x)|f(x)$  and  $d(x)|g(x)$
23.             go to step 27;
24.         else
25.             set  $D = \deg e_p(x)$ ;
26.             go to step 12.
27. Output the result:  $\mathbf{gcd}(f(x), g(x)) = r \cdot d(x)$ .

EXAMPLE 8.1. Let us apply Algorithm 8.1 to polynomials (1) from Knuth's example. Since  $w = 1$ , take  $q = 2$ . We have already computed in Example 7.2 that  $e_2(x) = \mathbf{gcd}(f_2(x), g_2(x)) = x^2 + x + 1$ . Then  $s(2, f, g) = \deg e_2(x) = 2$  and

$$M_{2,f,g} = 2^2 \cdot 1 \cdot \min \left\{ \frac{\sqrt{113}}{1}, \frac{\sqrt{570}}{3} \right\} < 31.84.$$

Take  $p = 67 > 2 \cdot M_{2,f,g}$ . It is easy to calculate that  $\mathbf{gcd}(f_{67}(x), g_{67}(x)) \approx 1$ . Compare this with Example 5.2, where we had to use much larger prime  $p = 1031$ . Moreover, if we take as an auxiliary  $q$ , say,  $q = 3$ , then  $s(3, f, g) = \deg e_3(x) = 0$  and  $M_{3,f,g} \leq 15.92$ . So we can take an even smaller prime  $p = 37 > 2 \cdot M_{3,f,g}$ .

The ideas of Algorithm 7.1 and of Algorithm 8.1 can be combined to work with more than one auxiliary prime  $q$ . Like we mentioned in Remark 7.1, Knuth in [7] recommends to first check if the polynomials  $f(x), g(x) \in \mathbb{Z}[x]$  are coprime, and to proceed to their  $\mathbf{gcd}$  calculation only after we get that they are not coprime (this is motivated by probabilistic arguments). Compute  $A_{f,g}$  by formula (18) and find a  $k$  like

we did it in step 09 of Algorithm 7.1:  $k$  is the maximal number for which  $p_k\# \leq A_{f,g}$ . Then choose any  $k+1$  primes  $q_1, \dots, q_{k+1}$  not dividing  $w$ , and start computing the modular **gcd**'s  $e_{q_1}(x), e_{q_2}(x), \dots$  ( $k+1$  times). If at some step we find  $\deg e_{q_i}(x) = 0$ , then we are done: the polynomials  $f(x), g(x)$  are coprime if  $r \approx 1$ , or their **gcd** is the non-trivial scalar  $r \not\approx 1$ . And if  $\deg e_{q_i}(x) > 0$  for all  $q_i$ , then we know that:

- (1) these polynomials are not coprime, and
- (2) the positive degree of **gcd**( $f(x), g(x)$ ) is the minimum

$$(22) \quad s(f, g) = \min \{ \deg e_{q_1}(x), \dots, \deg e_{q_{k+1}}(x) \} > 0.$$

This exact value of  $s(f, g) = \deg \mathbf{gcd}(f(x), g(x))$  is a better result than the estimate  $s(q, f, g)$  obtained earlier by just one  $q$ .

Like above, we can assume  $f(x), g(x)$  to be primitive (if not, we can again denote  $r = \mathbf{gcd}(\text{cont}(f(x)), \text{cont}(g(x)))$  and switch to the primitive parts  $f(x) = \text{pp}(f(x))$  and  $g(x) = \text{pp}(g(x))$ ). Applying the Landau-Mignotte Theorem 4.2 for the coefficients  $c_i$  of **gcd**( $f(x), g(x)$ ), we get that  $|c_i| \leq M_{f,g}$ , where

$$(23) \quad M_{f,g} = 2^{s(f,g)} \cdot \mathbf{gcd}(a_0, b_0) \cdot \min \left\{ \frac{\|f(x)\|}{|a_0|}, \frac{\|g(x)\|}{|b_0|} \right\}.$$

Now we can take a  $p > 2 \cdot M_{f,g}$  and by the Euclidean algorithm calculate  $e_p(x)$  in  $\mathbb{Z}_p[x]$ . If  $\deg e_p(x) > s(f, g)$ , we drop this  $p$  and choose another prime  $p > 2 \cdot M_{f,g}$ . And if  $\deg e_p(x) = s(f, g)$ , then we proceed to the final steps: we choose the  $t$ , then get the preimage  $k(x)$  of  $t \cdot e_p(x)$ , then go to the primitive part  $d(x) = \text{pp}(k(x))$  and output the final answer as **gcd**( $f(x), g(x)$ ) =  $r \cdot \text{pp}(k(x))$ .

**REMARK 8.2.** This approach has the following advantages: Firstly, the bound on primes  $p$  is better than formula (21) since here we have not  $2^{s(q,f,g)}$  but  $2^{s(f,g)}$ . Secondly, we no longer need calculate the number  $t$ , the preimage  $k(x)$  and the primitive part  $d(x)$  for *more than one* prime  $p$ . Because, if the selected  $p > 2 \cdot M_{f,g}$  is not appropriate, we already have an indicator of that:  $\deg e_p(x) > s(f, g)$ .

We built the following algorithm:

**ALGORITHM 8.2** (Big prime modular **gcd** algorithm with preliminary estimates on divisor degrees by multiply primes).

*Input:* non-zero polynomials  $f(x), g(x) \in \mathbb{Z}[x]$ .

*Calculate their greatest common divisor* **gcd**( $f(x), g(x)$ )  $\in \mathbb{Z}[x]$ .

01. Calculate  $\text{cont}(f(x)), \text{cont}(g(x))$  in Euclidean domain  $\mathbb{Z}$ , choose their signs so that  $\text{sgn cont}(f(x)) = \text{sgn lc}(f(x))$  and  $\text{sgn cont}(g(x)) = \text{sgn lc}(g(x))$ .
02. Set  $f(x) = \text{pp}(f(x))$  and  $g(x) = \text{pp}(g(x))$ .
03. Compute the bound  $A_{f,g}$  for polynomials  $f(x), g(x)$  by (18).
04. Find the maximal  $k$  for which  $p_k\# \leq A_{f,g}$ .
05. Calculate  $r$  in Euclidean domain  $\mathbb{Z}$  by (16).
06. Set  $a_0 = \text{lc}(f(x))$  and  $b_0 = \text{lc}(g(x))$  (they are positive by our selection of signs for  $\text{cont}(f(x))$  and  $\text{cont}(g(x))$ ).
07. Calculate the positive  $w = \mathbf{gcd}(a_0, b_0)$  in Euclidean domain  $\mathbb{Z}$ .

08. Set  $s(f, g) = \min\{\deg f(x), \deg g(x)\}$ .
09. Set  $i = 1$ .
10. While  $i \neq k + 1$ 
  11. choose a new prime  $q_i \nmid w$ ;
  12. apply the reduction  $\varphi_{q_i}$  to calculate the modular images  $f_{q_i}(x), g_{q_i}(x) \in \mathbb{Z}_{q_i}[x]$ ;
  13. calculate  $e_{q_i} = \mathbf{gcd}(f_{q_i}(x), g_{q_i}(x))$  in Euclidean domain  $\mathbb{Z}_p[x]$ ;
  14. if  $\deg e_{q_i} \leq s(f, g)$ 
    15. set  $s(f, g) = \deg e_{q_i}$ ;
    16. if  $\deg e_{q_i} = 0$ 
      17. set  $d(x) = 1$ ;
      18. go to step 32;
  19. set  $i = i + 1$ .
20. Calculate  $M_{f,g}$  by (23) using the value of  $s(f, g)$ .
21. Choose a new prime number  $p > 2 \cdot M_{f,g}$ .
22. Apply the reduction  $\varphi_p$  to calculate the modular images  $f_p(x), g_p(x) \in \mathbb{Z}_p[x]$ .
23. Calculate  $e_p(x) = \mathbf{gcd}(f_p(x), g_p(x))$  in Euclidean domain  $\mathbb{Z}_p[x]$ .
24. If  $\deg e_p = s(f, g)$ 
  25. choose a  $t$  such that the  $\text{lc}(t \cdot e_p(x)) = w$ ;
  26. call Algorithm 4.1 to calculate the preimage  $k(x)$  of  $t \cdot e_p(x)$ ;
  27. calculate  $\text{cont}(k(x))$  in Euclidean domain  $\mathbb{Z}$ ;
  28. set  $d(x) = \text{pp}(k(x)) = k(x)/\text{cont}(k(x))$ ;
  29. go to step 32;
30. else
  31. go to step 21.
32. Output the result:  $\mathbf{gcd}(f(x), g(x)) = r \cdot d(x)$ .

EXAMPLE 8.3. Let us apply Algorithm 8.2 again on polynomials of Knuth's example (1). As we saw in Example 6.6,  $k = 30$ . So we may have to consider at most 31 auxiliary primes  $q_i$ . But we in fact need just two of them, because  $\deg \mathbf{gcd}(f_2(x), g_2(x)) = \deg(x^2 + x + 1) = 2$  and  $\deg \mathbf{gcd}(f_3(x), g_3(x)) = \deg(1) = 0$  (see Example 7.2). So in Algorithm 8.2 we jump from step 16 to step 32 directly.

Vahagn H. Mikaelian:  
 Informatics and Applied Mathematics Department  
 Yerevan State University  
 Yerevan 0025, Armenia.  
 E-mail: v.mikaelian@gmail.com

## REFERENCES

- [1] W. Brown, *On Euclid's Algorithm and the Computation of Polynomial Greatest Common Divisors*, J. ACM, 1971, 18, 478–504.
- [2] P. M. Cohn, *Basic algebra. Groups, rings and fields*, Springer-Verlag, London, 2003.
- [3] P. M., Cohn *Introduction to ring theory*, Springer-Verlag, London, 2000.

- [4] J. H., Davenport, Y. Siret, E. Tournier, *Computer algebra. Systems and algorithms for algebraic computation*, Second edition Academic Press, London, 1993.
- [5] P. B. Garrett, *Abstract algebra*, Boca Raton: Chapman & Hall/CRC, 2008.
- [6] D. S. Dummit, R. M. Foote, *Abstract algebra*, Third edition s.l.: John Wiley and Sons, 2004.
- [7] D., Knuth, *The art of computer programming. Vol. 2. Seminumerical algorithms.*, Second edition. Addison-Wesley Series in Computer Science and Information Processing. s.l.: Addison-Wesley, 1969.
- [8] A. I. Kostrikin, *An introduction to algebra. Vol. 2. Main structures.*, FizMatLit, Moscow, 2004 (Russian).
- [9] M. Mignotte, *Mathematics for computer algebra. Translated from the French by Catherine Mignotte*, Springer-Verlag, New York, 1992.
- [10] S. Lang, *Algebra*, Revised third edition. Graduate Texts in Mathematics, 211.. New York: Springer-Verlag., 2002.
- [11] V. H. Mikaelian, *An introduction to computer algebra*, in preparation.
- [12] V. H. Mikaelian, *The Big prime gcd algorithm on general UFD's*, in preparation.
- [13] V. H. Mikaelian, *Metabelian varieties of groups and wreath products of abelian groups*, J. Algebra, 2007 (313), 2, 455–485.
- [14] E. V. Pankratev, *Elements of computer algebra Study Guide*, BINOM. LZ, Moscow, 2007.
- [15] J. von zur Gathen, J. Gerhard *Modern Computer Algebra*, Third edition, Cambridge University Press, Cambridge, 2013.